NASA Technical Memorandum 87271

A Sensor Failure Simulator for Control System Reliability Studies

Kevin J. Melcher, John C. Delaat, Walter C. Merrill,
Lawrence G. Oberle and Gerald G. Sadler
Lewis Research Center
Cleveland, Ohio

and

Joseph H. Schaefer
United States Corps of Cadets
West Point, New York

July 1986

NASA

A SENSOR FAILURE SIMULATOR FOR CONTROL SYSTEM RELIABILITY STUDIES

Kevin J. Melcher, John C. Delaat, Walter C. Merrill,
Lawrence G. Oberle, and Gerald G. Sadler
National Aeronautics and Space Administration
Lewis Research Center
Cleveland, Ohio 44135

and

Joseph H. Schaefer
United States Corps of Cadets
West Point, New York

SUMMARY

A real-time Sensor Failure Simulator (SFS) was designed and assembled for the Advanced Detection, Isolation, and Accommodation (ADIA) program. Various designs were considered. The design chosen features an IBM-PC/XT. The PC is used to drive analog circuitry for simulating sensor failures in real-time. A user defined scenario describes the failure simulation for each of the five incoming sensor signals. Capabilities exist for editing, saving, and retrieving the failure scenarios. The (SFS) has been tested closed-loop with the Controls Interface and Monitoring (CIM) unit, the ADIA control, and a real-time F100 hybrid simulation. From a productivity viewpoint, the menu driven user interface has proven to be efficient and easy to use. From a real-time viewpoint, the software controlling the simulation loop executes at greater than 100 cycles/sec.

INTRODUCTION

This report describes a general purpose device which can simulate sensor failures in control systems. This device, called the SFS, is personal computer based, programmable, reliable, and flexible. It also provides repeatable failure simulations. With these features the SFS can be used to efficiently evaluate and demonstrate sensor failure detection logic. The SFS interface includes five separate analog signal flow paths through the device with independent digital control of modifications (i.e., sensor failures) made to the analog signals. The first application of the SFS is simulation of sensor failures for the Advanced Detection, Isolation, and Accommodation (ADIA) program (ref. 1).

The goals of the ADIA program are to develop, implement, evaluate, and demonstrate an ADIA algorithm. The development and real-time implementation of the algorithm are described in reference 1. Algorithm performance was evaluated using a real-time hybrid computer simulation of the F100 engine, the sensor failure simulator, and the ADIA control (ref. 2). Finally, the algorithm will be demonstrated on a full scale Pratt and Whitney F100 engine in the Lewis Research Center's altitude facility. Both the ADIA control and the sensor failure simulator will be used in this demonstration.

This report describes the SFS which was developed for the ADIA program. Included is a discussion of the design requirements as well as system concept and philosophy. This is followed by a description of the hardware and the software design. Finally a guide to operations for the simulator is given.

REQUIREMENTS

The SFS was developed for the ADIA program. The ADIA control currently uses signals from five sensors: fan shaft speed (N1), compressor shaft speed (N2), combustor exit pressure (PT4), low turbine exit pressure (PT6), and low turbine inlet temperature (FTIT). The SFS was designed to modify any combination of the five ADIA sensor signals and send the modified (or failed) signals to the ADIA control system under experimental test conditions. In order for the SFS to properly perform this task, certain initial design requirements had to be met.

The first requirement for the SFS was that it must model failures according to the following equation:

$$y_{out} = (\text{scale factor} \cdot y_{in}) + \text{bias} + \text{noise}$$

This equation describes a failure as the sum of: a scale factor multiplication of the incoming sensor signal, a bias (step + ramp), and random white noise. By modeling failures in this manner (fig. 1), the SFS should allow a user to simulate most of the types of failures observed in engine sensors.

The next requirement imposed on the SFS was an ability to maintain signal integrity. The signals leaving the SFS must be the same as the in-coming signals in a normal/unfailed mode. If a discrete system is chosen, there should be no significant sampling delay. Safety is also an important consideration. A device failure, such as loss of power to the SFS, must not disrupt signal flow in the normal mode.

It was also required that the SFS be a stand alone, portable unit. The simulator will be required to perform its task in several facilities. It must first be located in the hybrid simulation facility at Lewis where it will be validated. After validation in the hybrid lab, the simulator and control will be moved to the Propulsion Systems Laboratory at Lewis. In this facility the SFS will be used to test the ADIA control on a full scale F100 engine. It is desirable that no disassembly/assembly need take place during this transition.

Another requirement was that the SFS have a convenient user interface with reasonable programmability. The SFS should be simple to use and it should provide for a high degree of productivity in an environment where overhead costs are substantial. The time required to prepare a failure scenario between data points should be minimal.

Finally, the SFS should demonstrate reliability, maintainability, predictability, and repeatability. The SFS should be reliable, having a high mean time between failures. Safety is a major concern when testing a full-scale engine. A reliable simulator will be necessary to limit the risk involved when testing the ADIA control with the F100 engine. During testing, engine sensor signals will be failed intentionally to check the fault accommodation and

detection of the control. These actions may be catastrophic to the engine, especially if valid engine sensor signals are not available at all times. For these reasons, the design must meet the safety requirements of the Lewis safety committee. Also, overhead costs tend to be high in an experimental environment. Therefore, it was desirable to choose a design which could be easily maintained, thus reducing downtime. A modular design would meet this requirement. Predictable and repeatable performance is also necessary as a basic characteristic for a good research tool.

As a final requirement, development time for the SFS must be within the constraints of the ADIA program schedule. This would suggest a design which would be based primarily on commercially available hardware.

SYSTEM CONCEPT AND PHILOSOPHY

Various conceptual designs were suggested for the SFS. In all of the designs a common underlying philosophy was evident. This philosophy addresses the failure modeling and the signal integrity requirements by combining a failure scenario controller with a direct analog signal path (fig. 2). The general concept is to model failures using analog circuitry and to determine the size and timing of the scale factor, bias or noise failure components using the scenario controller. It was decided early in the design process that digital sample and hold hardware could not be permitted in the direct signal flow path for both safety and performance reasons.

To ensure signal integrity during normal operation or in the event of a loss of power to the SFS the following approach was adopted. Failure simulation is initiated by a relay contract closure and terminated in the same manner. The normally open relay contracts allow a direct, uninterrupted, signal path during normal unfailed operation.

Four possible designs were suggested and studied for their ability to meet the specified design requirements. The four designs were: (1) custom microcomputer driven analog hardware, (2) personal computer driven analog hardware, (3) analog computer driven analog hardware, and (4) programmable controller driven analog hardware.

Each of the designs was considered for its ability to meet the simulator design requirements. The custom microcomputer based design met all requirements except for development time constraints. It was not possible to build and test the microcomputer design within scheduled deadlines. The personal computer-based design met all of the design requirements. Required digital to analog interface hardware was available "off the shelf." Custom development would include the software, and analog summing circuit, and a communications circuit for interfacing the PC with the Control Interface and Monitoring (CIM) unit (ref. 3). It was determined that this development could be accomplished within the necessary time constraints. The strictly analog design had several deficiencies. The analog computer tends not to be user friendly and available hardware is not portable. Also repeatability and reliability are difficult to obtain. The programmable controller based design was also determined to be undesirable. The available programmable controller was designed for processes with fixed logic. It was not designed to allow for program changes during execution.

Based on the above requirements analysis, the PC/analog design was chosen for the SFS. This design met all of the stated design requirements. Additionally, this design has the flexibility and generality to be used in other failure detection studies and/or allow simulation of various other failure models.

HARDWARE DESIGN

The design chosen for the SFS was based on a personal computer (PC) interface/controller driving analog signal processor hardware. The PC used for the SFS is a standard configuration IBM-PC/XT expanded to 640K bytes of memory. An AST Six Pack/Plus expansion board was used for memory expansion. The AST board also contains a clock which is used by the SFS software. An expansion chassis with a PC interface houses most of the analog failure circuitry. This circuitry is described in detail throughout the remainder of this section.

Figure 3 is a block diagram of the SFS hardware design. Three general observations can be made about this design. First, the five engine sensor signals are direct inputs to the normally closed terminals of the ERB-24A switch matrix. In the normal/unfailed mode, each of the engine signals completely bypasses the SFS and proceeds through the common terminals of the switch matrix to the ADIA control. Second, the simulator may modify any number of the five sensor signals by adding scale factor or bias errors to the original signal. A noise error may also be imposed on the any of the five sensor signals, however it may be added to only one channel at a time. These modified signals are fed to the normally open (NO) terminals of the switch matrix. The computer may then select either a modified, or an unmodified signal for each of the ADIA controller's five inputs. Third, the five engine signals are electrically differnetial with no reference ground. The modified signals must be compatible in order to replace the unmodified signals. The simulator signals are transformed into virtual-differential signals in the circuitry which combines the contributions of the three error components: scale factor, bias, and noise. These components are each produced in slightly different ways.

The scale factor error, or multiplication of the incoming signal by a constant, is generated by using a METRABYTE DAC-02 multiplying digital-to-analog converter (MDAC) for each channel. The MDAC has two inputs, an analog signal (in this case, one of the five engine signals) and a digitally encoded number which the MDAC receives from the PC. The MDAC produces an analog voltage output equivalent to the product of its inputs. Each DAC-02 circuit board contains two MDACs. Thus, to cover the five engine signals, three DAC-02's must be employed.

The bias errors for the five signals are generated using a METRABYTE DDA-06 digital-to-analog converter (DAC). The DAC receives a digitally encoded number from the computer representing the amount of bias to be added to the in-coming analog signal, and generates an analog voltage, in the range ±10 V, proportional to this number. The DDA-06 provides six such DAC's, which leaves one spare channel for future use.

The noise error is generated by a commercially available analog random noise generator. The output from the noise generator is scaled using the spare MDAC from the scale factor circuitry. The output from the MDAC is then switched to any one of the five engine signals using spare relay channels on

the switch matrix. These spare relay channels are labelled ERB-24B in figure 3. Both the switch matrix and the MDAC receive their commands from the computer. Since only one noise signal is generated, only one of the five engine signals can be modified using all three error components, at any one time. The other four signals can be provided with any combination of scale factor, and biasing errors.

For each of the five signals, the modified engine signal is generated by summing contributions from the scale factor, bias and noise error components; using an analog summing circuit, as shown in figure 4. This circuit was replicated exactly for each of the five channels. It is a standard design, using Zener diodes to limit the summed voltages to ±10 V maximum; and providing a virtual-differential output voltage to the switch matrix. The second op-amp in figure 4 is provided so that the summation is not inverted.

The DACs, MDACs and a custom analog signal processor board are found in the expansion chassis. A layout of the expansion chassis is shown in figure 5. There are two unused card slots in the expansion chassis, but only one unused connector location, since the analog signal processor (ASP) board requires two connectors.

A block diagram of the ASP board is shown in figure 6. At connector AJ1 are the ten lines representing the five differential input engine signals. Each pair of these lines is an input to a Burr-Brown 3630 Instrumentation Amplifier with unity gain. The resulting output, a single-ended signal is one of five output signals at connector AJ2. Also at connector AJ2 are 15 lines (five triplets) representing signals from the three failure components. These signals for scale factor error, bias error, and random noise are summed by one of the five summing circuits as described previously. The five resulting virtual-differential modified engine signals are then wired via ten lines to connector AJ1. The ASP board contains the components and wiring for the five instrumentation amplifiers, and the five summing and isolation amplifiers.

The ASP board also contains circuitry for a communications interface between the Control Interface and Monitoring (CIM) unit and the SFS. This application specific hardware is provided as a means of synchronizing the beginning of failure simulation with the beginning of data taking in the controls computer. A D/A converter on the controls computer is used to send a start signal to the SFS interface circuitry (fig. 7). On the ASP board, the start signal is first converted from a differential signal to a single-ended signal using an instrumentation amplifier. A comparator is then used to detect when the start signal is high and to convert it to TTL levels. Finally, the output of the comparator is sent to Port C of the 8255 chip on the DDA-06 board where it can be detected by the SFS software.

The switch matrix chosen for this simulator is the METRABYTE ERB-24. The ERB-24 provides 24 channels of double pole/double throw relays. Ten of these relays are used by the SFS. Of all the components selected, the ERB-24 is the only one which does not reside on the PC bus. Due to its size this component required a separate chassis. All interface wiring is done inside this chassis. The end panel for the ERB-24 chassis is shown in figure 8. The eight connectors labeled AJ1, AJ2, MJ1, MJ2, MJ3, EI1, DJ1, and EO1 correspond to the two connectors for the ASP board, the connectors for the three DAC-02 MDAC boards, the engine signal input connector, the connector for the DDA-06 DAC board, and

the engine signal output connector, respectively. The pin assignments for these connectors are shown in tables I to VI.

SOFTWARE DESIGN

The SFS software was conceived as a menu driven program which would provide four distinct capabilities: on-line program instruction, storage and retrieval of failure scenarios, editing of failure scenarios, and real-time control of the analog failure simulation hardware. The instruction capability was to be designed as a means of providing operational instructions during program execution. These instructions should be both general and specific so that the SFS would be as self contained as possible. The store/retrieve capability was to allow the user to store and retrieve pre-defined failure scenarios. This capability would help provide the repeatability and high productivity so desired in a research environment. The means of defining and modifying failure scenarios were provided by a failure scenario editor. This editor was to provide an efficient and user-friendly method of modifying any or all of the components which combine to form a failure scenario: the scenario description, the failed/unfailed channels, the nominal and maximum channel values, the failure delay for each channel, and the constants associated with the scale factor, step, ramp, and noise failure modes. The simulation portion of the software was assigned the task of real-time failure simulation based on information contained in any given failure scenario. This task was to include software which would scale failure scenario parameters, initialize the analog hardware, and begin execution of real-time failure simulation based on a cue from the CIM unit.

Although not in the original conceptual design, a fifth task was added to the SFS software during development. This task provides the user with the ability to trip relays or to set D/A constants directly from the keyboard. This task was included to facilitate debugging of the hardware and software.

Once the general conceptual design for the SFS software was established (fig. 9), the next step was to choose a method of implementation which would provide for a highly efficient user interface. A menu driven approach was chosen. This type of approach is very efficient if the program execution can be described as a type of decision tree with a limited number of branches at any node. This was the case for the SFS as it was conceived and this was the approach taken during implementation.

The SFS currently makes use of 15 menus and numerous other prompts to accomplish its tasks. The user interface has proven to be very efficient and user-friendly. The scenario store/retrieve capability, while slightly cumbersome, provides the user with readable, as well as retrievable, scenario descriptions. The editor is easy to use, and very efficient due to the extensive use of function keys for input. The transient capability is able to provide the desired resolution during simulation, 2 to 9 msec. And the test capability has already proven useful in debugging both hardware and software.

Since the sensor failure simulator hardware was being implemented with an IBM-PC/XT as the user interface, a standared PC language was required to implement the software. The primary language chosen for implementation of the SFS software was FORTRAN. In particular, Ryan-McFarland Corporation's IBM Professional FORTRAN (version 1.00) compiler was used to produce the executable

code. This version "is an implementation of the full standard ANSI X3.9-1978 with extensions" (ref. 4). These extensions include utilities for obtaining the date and time from the Disk Operating System (DOS) clock.

As a secondary language, IBM Macro Assembler was chosen (ref. 5). FORTRAN does not have inherent in it the ability to interface with real-time hardware. Therefore it was necessary to write some assembler code for driving the analog circuitry and for interfacing with the CIM unit. To obtain higher resolution than was available from the DOS clock, it was also necessary to write an assembly language routine that would read the real-time clock on the AST memory expansion board. The 1 msec resolution available from the AST clock yields smooth failure transients relative to the control update cycle.

The SFS source code, about 8000 lines, is currently divided into 47 DOS text files which occupy approximately 182 Kbytes of hard disk storage. The FORTRAN code is contained in 42 files; one file for the main program (SFS.FOR), 33 files for subprograms, and nine files for common blocks. Common blocks are stored in separate files and included by the compiler during compilation. Four of the remaining five files contain subprograms written in Macro Assembler which provide hardware to hardware and software to hardware interfaces. The last "source file" is the file that contains the text for on-line instructions. A subprogram source file name is designated as the first eight letters of the subprogram's name. The extension ".FOR" is used to denote a file containing FORTRAN source code; the extension ".CMN" is used to denote a file containing a common block, and the extension ".ASM" is used to denote a Macro Assembler source code file. Table VII lists the source code file names, the name of the subprogram contained in each file, and a brief description of each subprogram. Table VIII lists the names of files containing common blocks, the name of the common contained in each file, and brief description of what is stored in the common. Table IX shows the hierarchical structure of the SFS program. No attempt has been made to show multiple calls.

The FORTRAN and the assembly language subprograms are compiled and assembled respectively. The object code produced by compilation of the main program is stored in SFS.OBJ. Object code for the subprograms is stored together in a single object library, SFS.LIB. This is accomplished by using the library utility supplied with the Professional FORTRAN, version 1.10 of the IBM Library Manager. Executable code is produced by using version 2.3 of the IBM Personal Computer Linker (ref. 4) and is stored in SFS.EXE.

The object code for the SFS is contained in two files which require a total of approximately 182 Kbytes of hard disk storage. The object module of the man program, SFS.OBJ, requires about 5 Kbytes. The rest of the 182 Kbytes is used for the object library, SFS.LIB.

The last two files which are part of the SFS code are the file containing the executable image, SFS.EXE, and the DOS text file containing instructions for using the SFS, INSTRUCT.TXT. The executable code requires 147 Kbytes of storage and the instruction file about 57 Kbytes.

GUIDE TO OPERATION

This section is designed as a user guide for the SFS. In it the operation of the SFS is described in detail. The first part of this discussion will deal

with directory and file structures. The later part of the section will discuss in-depth each of the various program menus.

It should be pointed out at this time that only one failure scenario at a time resides in the PC's random access memory (RAM). This failure scenario is called the current failure scenario. In most cases, the program will be working on/with the current failure scenario. A default scenario is defined during the program initialization and becomes the current failure scenario until changed by the user or replaced by the retrieval of a stored scenario.

It should also be pointed out at this time that some of the keys on the keyboard are "mapped" during execution. FORTRAN does not provide an interrupt capability for reading the keyboard. As a result some of the keyboard keys are mapped or redefined to provide useful capabilities like one-stroke input and use of function keys. A more in-depth discussion on this topic will be presented later in this paper. At this time the user is to be warned about terminating program execution in an irregular manner. The effects of the key mapping are such that, if execution is terminated irregularly, it may be necessary to reboot the PC to return all keys to their original key codes.

Directory and File Structure

Three files are used to run the SFS program. These files are SFS.BAT, SFS.EXE, and INSTRUCT,TXT; SFS.BAT resides in the directory \UTILITY on the PC's hard disk and is the batch file which initiates program execution. The second file mentioned above, SFS.EXE, contains the executable code and should be located in the directory \SFS. Also residing in the \SFS directory on the hard disk, is the DOS text file, INSTRUCT.TXT, which contains text for on-line instructions.

Although the SFS program and instructions are currently loaded from the hard disk drive, they may also be loaded from a floppy disk. In either case the directory and file structure should be the same as described above with the exception that the SFS.BAT file may be stored in the floppy's root directory.

Initiating Program Execution

Initiate program execution by typing "SFS" in response to the DOS prompt from anywhere but the \SFS directory. To get a printer listing of stored scenario files (.SFS extension) before initiating program execution, type "SFS /D" in response to the prompt.

Upon receiving the "SFS" command, DOS begins executing statements from the SFS.BAT batch file (table XI). In lines 1 and 2, the ECHO and BREAK are turned off. In line 3, the DOS search path is defined so that the search for an executable file proceeds from the current directory to the \SFS directory. Line 4 of the batch file changes the DOS default directory to the \SCENARIO directory where failure scenario files may be stored. Lines 5 and 6 check for the "/D" parameter. If it has been included in the call to the batch file, a directory

listing of all stored scenario files is routed to the printer. The SFS executable code is loaded and program execution begins after the DOS command processor receives line 7. For further explanation of these commands see the DOS manual (ref. 7).

The SFS program begins by printing a title screen. This screen is displayed while program initialization takes place. After initialization is completed, this screen is replaced by the program's main menu (fig. 10).

The main menu presents the user with a list of six options: (1) instructions, (2) retrieve stored failure scenario, (3) edit current failure scenario, (4) run current failure scenario, (5) test SFS hardware/software, and (6) quit. There are two ways to choose an action item from this menu. One way is to enter the number associated with a desired action item. Another way is to choose the default action item.

The default is displayed in reverse video. The default may be changed by pressing the up arrow or the down arrow on the keyboard. It may also be changed by entering the codes corresponding to these keys, "u<CR>" and "d<CR>" respectively. The default may be selected by the carriage return/enter key.

Instructions

The first action item in the SFS main menu provides the user with an on-line program reference. When the user chooses this action item, instruction text is displayed. The first page of this text is shown in figure 11. The user may page up and down through the text by pressing the PgUp and PgDn keys. Entering the code "b<CR>" of "f<CR>" will have the same effect. Other keys which may be used while in the instruction facility are the Home and End keys. The Home key causes the instruction facility to return to the first page of text. The End key causes the facility to proceed to the last page of text. The ASCII codes for these keys are "Home<CR>" and "End<CR>" respectively. The user may exit the instruction facility by pressing a carriage return/enter.

The text for the instruction facility is stored in the DOS text file INSTRUCT.TXT. It is stored as a series of pages 80 columns wide by 22 lines long. Macros are provided for including any ASCII character or sequence of characters in this text. These macros cause ASCII codes to be inserted in each page of text as it is read from the test file. Macros are listed in table XII. Note that while the macros are multiple characters, the number of characters displayed by the monitor will depend on the ASCII character or sequence of characters which define a given macro. Also, note all DOS control sequences must be followed by a space in the test. In particular, this applies to the $f, $b, $r, and $n codes.

Storing and Retrieving Failure Scenarios

The second action item in the SFS main menu provides the user with capabilities for storing and retrieving failure scenarios from disk storage. When the user chooses this action item the stored scenario menu is displayed (fig. 12).

The stored scenarios menu presents the user with five action items: (1) RETRIEVE, (2) DELETE, (3) REPLACE, (4) STORE, and (5) RETURN. These items may be chosen in a manner similar to the SFS main menu. Note that the user may wish to return to the stored scenario menu without exercising the specified action item; this may be accomplished by pressing the PgUp key or the PgDn key any time after an action item has been selected.

Action item number one allows the user to retrieve, from a specified stored scenario file, any scenario which has been stored in that file. A stored scenario file is any DOS text file in which only scenarios are stored or will be stored. One convention is to use a .SFS extension for denoting a failure scenario file. When action item one is chosen, the user is prompted for a file name. The default file name may be selected by pressing carriage return/enter or a new file name entered by the user. File names may be any valid DOS file name. If the file exists, the program will read the description of any scenario stored in this file and present the user with a list of the description(s) (fig. 13). The user may then specify, by setting the default, which scenario the program should RETRIEVE. When the specified scenario has been retrieved, the program displays the new scenario description under the banner and returns to the stored scenario menu.

Action item number two allows the user to delete, from a specified stored scenario file, any scenario which has been stored in that file. When action item two is chosen, the user is prompted for a file name. As with action item one, the program will check for the files existence. If the file exists, the program will read the description of any scenario stored in this file and present the user with a list of these descriptions. The user may then specify, by setting the default, which scenario the program should DELETE. When the specified scenario has been deleted, the program returns to the stored scenario menu.

Action item number three allows the user to replace a stored scenario with the current failure scenario. When this action item is chosen, the user is prompted for a file name. As with action items one and two, the program will check for the files existence. If the file exists, the program will read the description of any scenario stored in this file and present the user with a list of these descriptions. The user may then specify, by setting the default, which scenario should be REPLACED by the current failure scenario. When the specified scenario has been replaced, the program returns to the stored scenario menu.

Action item number four allows the user to store the current failure scenario in a specified file. When this action item is chosen, the user is prompted for a file name. The program will check for the files existence. If the file exists and the specified file has the capacity, the current failure scenario will be appended to the end of the file. There is a limit of ten (10) failure scenarios per scenario file. If the file specified by the user does not exist, the program notifies the user and asks if it should create a new file. When the scenario is stored, the user is returned to the stored scenario menu.

The user may return to the SFS main menu by selecting action item five.

Creating/Editing a Failure Scenario

The third action item in the SFS main menu is EDIT CURRENT FAILURE SCENARIO. When the user selects this action item, the program enters the FAILURE SCENARIO EDITOR.

The editor begins by displaying the description of the current failure scenario (fig. 14). At this time, the user may choose to accept the current description or replace it with a new description. Retention of the default description may be accomplished by pressing carriage return/enter. A new description may be defined by simply entering it from the keyboard.

After the user enters a description, the editor centers it below the banner and displays the Failed Channels menu (fig. 15). This menu allows the user to select which channels will be failed during simulation. Failed channels are displayed in bold type; unfailed channels are displayed in normal type. Each channel may be toggled between failed and unfailed by pressing the number key corresponding to the given channel. A carriage return notifies the program that the user is finished with this menu.

The next two menus in the failure scenario editor are necessary to establish the relationship between the SFS output voltages and the engineering units they represent. They allow the user to define the parameters of a failure scenario in engineering units. The first menu is for specifying nominal channel values (fig. 16). Each channel's nominal value should be set to the value, in engineering units, represented by the incoming sensor signal at run time. The second menu is for specifying maximum channel values (fig. 17). Each channel's maximum value should be set to the value, in engineering units, which represents full scale.

The constants associated with these menus may be changed in the following manner. First, set the default (reverse video) over the channel which will be modified. Use the up and/or down arrow key to set the default. Next, enter the new value of the constant followed by a carriage return. Pressing carriage return/enter by itself, causes the editor to proceed to the next menu. Note that any channels defined as failed in the Channel Failure menu will be displayed in bold text (high intensity) in these menus.

The failure delay menu (fig. 18) is the next menu displayed by the editor. It is functionally the same as the menus used to set nominal and maximum channel values. This menu allows the user to specify, for each channel, some dead time at the beginning of the transient. Actual failure simulation on each channel will begin immediately following the delay specified for that channel. This menu provides the user with the capability to simulate multiple offset failures.

After exiting the failure delay menu the editor will begin to display a menu of failure modes and constants (e.g., scale factor, bias, ramp, and noise) for each failed channel (fig. 19). Beginning with channel one and continuing sequentially through channel five, these menus allow the user to specify the failure modes and associated constants which define how a particular channel's failure will be simulated.

The menu for each channel failed will display four failure modes and four associated constants. Note that any combination of active/inactive modes are possible and that the active failure modes are displayed in bold type. A failure mode may be activated by simply depressing the associated number on the keyboard followed by a carriage return/enter. To modify the constant of a given failure mode, the user should enter the number corresponding to the failure mode followed by a delimiter (space or comma) followed by the new value of the constant. Pressing carriage return/enter completes the sequence and the constant's old value is replaced by the new.

In these menus the scale factor mode is always active and defaults to unity; the other modes may either be active or inactive. The active scale factor mode causes a channel's incoming signal to be multiplied by the scale factor constant. The value of scale factor constant may range between ±2. If the bias mode is active, it has the effect of adding a step to the incoming sensor signal. The height of the step is the value of the bias constant which is only constrained by the specified maximum channel value. If the ramp failure mode is active, a bias is added to the incoming sensor signal which varies linearly in time, the slope being the value of the ramp constant. If the noise mode is active, the incoming signal from the external noise generator is multiplied by the noise gain constant and added to the incoming sensor signal. The noise gain constant is limited to the range ±1. The noise failure is allowed to be active on only one channel in the current failure scenario.

A single carriage return/enter will cause the editor to check for errors in failure definition. First, the editor checks to see if the maximum channel value is exceeded by the scale factor, step bias, or noise failures. If the maximum value is exceeded, the editor presents the user with current maximum and a suggested maximum. The user is asked if it is acceptable to replace the current maximum with the suggested maximum (fig. 20). If the user chooses not to accept the computed maximum, the editor returns to the failure mode menu. This error checking takes place because the transient portion of the program limits the output signal to the maximum channel value in both the positive and negative directions.

After conflicts with the maximum channel value have been satisfactorily resolved, the editor does some checking on the ramp failure mode. If the ramp failure mode is active, the program will display the approximate time at which the ramp will reach the channel maximum (fig. 21). The user is then prompted to accept the status quo. If the peak time displayed is unacceptable, entering an "n<CR>" will cause the editor to return to the failure mode menu. If the peak time displayed is acceptable, the editor moves on to the menu for the next failed channel.

When menus for all failed channels have been completed, the editor performs one final error check. It was stated previously that noise may be defined on only one channel. The editor will check this condition. If noise has been defined on more than one channel, those channels are displayed in menu form (fig. 22). The user is then asked to choose a single channel from the list. During run time, noise will only be added to the specified channel.

When the editor determines that the scenario is essentially without error, it queries the user one last time. This query allows the user to store the failure scenario just defined. If the editor receives a positive reply, the user will be prompted for a file name. The program will check for the files

existence. If the file exists and the specified file has the capacity, the current failure scenario will be appended to the end of the file. Remember, there is a limit of ten (10) failure scenarios per scenario file. If the file specified by the user does not exist, the program notifies the user and asks if it should create a new file. When the scenario is stored, the user is returned to the SFS main menu.

At this time more discussion should be included about mapping keys to specified codes. Most of the key mapping mentioned earlier was implemented specifically for the editor. Some keys are mapped to SFS identifiable sequences of ASCII codes before entering a menu and remapped to the original single ASCII codes when exiting the menu. At other times during program execution, a carriage return is added to a key's ASCII code. This provides a capability for one stroke input (e.g., pressing key "1" becomes the same as pressing key "1" followed by a carriage return). Key mapping is accomplished by the control codes listed in reference 6. At any time during execution an SFS identifiable ASCII sequence may be entered as an alternative to pressing the key to which that sequence is mapped. The ten function keys, as well as the Home, End, PgUp, and PgDn keys, are mapped to control codes recognized by the editor. These codes allow the user the freedom of moving between nonconsective menus within the editor. This feature was added to enhance productivity in the research environment. Table X contains a list of the function and keypad keys recognized by the editor, the codes which are mapped to these keys, and a short description of the keys functions. Figure 23 shows the function key template.

Real-Time Sensor Failure Simulation

The fourth action item in the SFS main menu is the heart of the SFS. This is where the actual real-time sensor failure simulation takes place. When the user chooses this action item the program requests permission to initialize the D/A hardware (fig. 24). There are several places in this part of the program where the user may abort the simulation and return to the main menu; this is the first. If the user enters a character other than "Y" or "<CR>" the program returns to the main menu.

If permission is granted by the user, the D/A hardware is initialize as follows. First, the multiplying DAC for the noise, MDAC0, is set to 0.0. Second, the scale factor MDACs for the five sensor signals are set to 1.0. Third, the bias DACs for the five sensor signals are set to 0.0. Forth, if the noise failure is active for a given channel, the noise relay for that channel is closed and all other noise relays are opened. Finally, if any channel has been defined as failed, the failure relay for that channel is closed. At this point, the signal flow path of all failed channels is redirected through the failure circuitry. The hardware is ready to begin the simulation of sensor failure(s).

After completing hardware initialization, the program does some software initialization. The failure gains specified for each channel are scaled and stored as run-time gains for output to the D/A hardware. Bias and ramp constants for channels 1 to 4 are scaled as follows:

$$\text{run-time gain} = \frac{\text{failure gain}}{\text{maximum channel value}}$$

For channel five the maximum value must be converted, before scaling, from Fahrenheit to Rankin. The formula for scaling the bias and ramp constants then becomes:

$$\text{run-time gain} = \frac{\text{failure gain}}{(\text{maximum channel value} + 459.67)}$$

Constants for the scale factor and noise failures are not normalized.

When initialization of the failure hardware is complete and the run-time gains have been computed, the program queries the user once again (fig. 25). Here the user is asked to specify how long the failure transient should run. If a carriage return is depressed, the program uses the default displayed under the query. If a run time other than the default is desired, that number may be entered from the keyboard. The software currently limits the run time to between 1 and 60 sec. Any integer or real number within these constraints may be specified.

After obtaining the length of the failure transient from the user, the SFS is ready to run the current sensor failure scenario. The program now presents the user with three choices (fig. 26): (1) Begin Scenario, (2) Begin Scenario on signal from CIM, and (3) Return to Main Menu. The desired action item is specified by a two character sequence, the option number followed by a carriage return/enter. If the user chooses to return to the main menu, the failure relays are opened so that the sensors signals are restored to their individual through flow signal paths.

When the user chooses action item number one, three things happen: the menu is erased; the message "** RUNNING **" appears with flashing asterisks (fig. 27); the real-time failure simulation begins. There is a lag of about 100 msec between the time when the user enters the option number and the time that failure simulation actually begins. Most of this lag is cause by output to the monitor.

When the user chooses action item number two a different sequence of events takes place. First, the current menu is erased. Then, the program notifies the user that it is waiting for a signal from the CIM unit to begin the simulation. When transient data taking is initated by the CIM unit, it sends a signal to the SFS. The SFS takes this signal as its cue to begin the simulation. An asterisk "*" is displayed under the wait message just before and during the real-time simulation (fig. 28). The dead time between the CIM signal and the beginning of the transient is approximately 40 msec.

Simulation begins by initalizing the timer. After the timer is initialized, the program enters the simulation loop. The simulation loop begins by calling the timer. The timer returns the time, in seconds, that the simulation has been running (run time). From this time and the time read at the beginning of the previous loop a delta (dT) is computed. At this point, the program begins to modify the constants of the D/A hardware.

If a channel is failed, and if the run time has just met or exceeded the failure delay, the MDAC for the noise, the MDAC for the scale factor, and the DAC for the bias are all set to their failed values. Next, if the channel is failed and if a ramp failure has been specified, a new value is computed for

the constant which is output to the bias DAC. The new value for the constant is computed as follows:

$$\text{run-time bias} = \text{run-time bias} + (\text{slope} * dT)$$

The variable slope on the right-hand side of the equation is the ramp constant scaled by the maximum channel value. The value for the bias DAC constant is then limited to values between ±1. After computing the new value for the bias DAC, it is output to the DAC. This series of steps is performed on channels 1 through 5 sequentially.

At the end of the simulation loop, the program saves the current run time and uses it to compute the next update interval (dT). If the run time is less than the time specified as the length of transient, execution continues at the beginning of the loop. When the run time meets or exceeds the length of the transient, the program exists the real-time loop and displays some statistical information about the run (fig. 29).

Note that this method of implementation allows the simulation loop to run at the maximum speed of the PC. The dT will change with the number of operations performed inside the loop. The worst case scenario occurs when all five channels fail at T = 0.0 and all five channels exhibit scale factor, bias, and ramp failures. For this worst case scenario the statistics were found to be: maximum dT is 0.009 sec, maximum dT occurs at 0.001 sec, and average dT is 0.005 sec. These figures are provided as a measure of the resolution of the simulation.

The final screen displayed by this part of the program is a menu provided for manually restoring channels to an unfailed state (fig. 30). It is displayed only if a failure has been simulated on one of the channels. Failed channels are displayed in bold (high intensity) type. All channels MUST be restored to an unfailed state. A channel may be restored to its unfailed state by entering the channel number from the keyboard. This trips the channel's relay which causes the sensor signal to be switched from the failed signal path to the through flow signal path. When all channels have been restored to an unfailed state, the program returns to the main menu.

Testing the SFS Hardware

The fifth action item in the SFS main menu provides the user with the capability to set the D/A hardware constants and trip noise and failure relays manually. Specifying this action item causes the program to erase the main menu, to initialize the D/A hardware, and to present the user with a menu similar to figure 31. This portion of the SFS was useful for debugging problems with both hardware and software.

Before displaying the test menu, constants for the D/A hardware are obtained from the current failure scenario. These scaled values are then output to the DACs and MDACs. The position of the noise relays is also set according to the current failure scenario. However, the failure relays are ALWAYS initialized to an unfailed state.

At this point the menu is displayed. If a DAC or an MDAC is loaded with a nonzero constant, the item will be displayed in bold type (high tensity). When relays which are positioned to a failed state are also displayed as bold.

Constants may be checked or changed by entering a device number followed by a carriage return/enter. Following this sequence, the current value of the constant for the specified device is displayed in the lower left corner of the monitor. Next a prompt for the device's constant and the limits of that constant are displayed. If only a carriage return is entered, the value of the constant for the device in question remains unchanged. If a new value is entered by the user, this value is output to the proper device and displayed in the lower left corner of the monitor. Values which are out of range cause an error message to be displayed.

The user may exit the test menu and return to the SFS main menu by entering device number 99. Before returning to the main menu, the program returns all relays to an unfailed state.

TERMINATING EXECUTION OF THE SFS

The last action item in the SFS main menu is for terminating program execution. When the user chooses action item number six, the main menu is erased and the user is presented with the prompt "Exiting SFS: Are you Sure??." An affirmative reply from the user causes program execution to be suspended. A negative reply returns execution to the main menu. It is STRONGLY SUGGESTED that program execution be terminated in this fashion. Terminating the program in any other manner may leave failure hardware in an undesirable state. Improper mapping of keyboard keys is also likely to occur if the program execution is terminated in an other than proper manner.

SUMMARY OF RESULTS

A real-time Sensor Failure Simulator was designed and assembled for the ADIA program. A personal computer-based design was chosen as the most favorable approach. In this design special analog hardware, driven by an IBM-PC/XT, modifies five incoming sensor signals to produce simulated sensor failures.

A user defined scenario contains the information which is used by the SFS to simulate sensor failures. The model used for simulating sensor failures has three components: a scale factor component, a bias component (constant + variable), and a noise component. Capabilities exist for editing, saving, and retrieving the failure scenarios.

The Sensor Failure Simulator has been tested closed-loop with the CIM, ADIA control, and a real-time F100 hybrid simulation. From a productivity viewpoint, the menu driven user interface has proven to be efficient and easy to use. From a real-time viewpoint, the software controlling the simulation loop executes than 100 cycles/sec.

REFERENCES

1. Delaat, J.C.; and Merrill, W.C.: A Real-Time Implementation of an Advanced Sensor Failure Detection, Isolation and Accommodation Algorithm. AIAA Paper 84-0569, Jan. 1984.

2. Merrill, W.C.; and Delaat, J.C.: A Real-Time Simulation Evaluation of an Advanced Detection, Isolation and Accommodation Algorithm for Sensor Failures in Turbine Engines. NASA TM-87289, 1986.

3. Delaat, J.C.; and Soeder, J.F.: Design of a Microprocessor-Based Control, Interface and Monitoring (CIM) Unit for Turbine Engine Controls Research. NASA TM-83433, 1983.

4. IBM Personal computer Professional FORTRAN, Installation and Use. Ryan-McFarland Corp., IBM Corp., 1984.

5. Macro Assembler Version 2.00. IBM Corp., 1984.

6. Disk Operating System, Technical Reference. Microsoft Corp., IBM Corp., 1983.

7. Disk Operating System. Microsoft Corp., IBM Corp., 1983.

APPENDIX A

TABLES

TABLE I. - AJ1 PIN CONNECTIONS -
PROTOBOARD DIFFERENTIAL SIGNALS

Pin	To	Function
1	DJ1-22	CIM READY
2	EI1-15	Channel No. 2 in (+)
3	EI1-16	Channel No. 2 in (-)
4	EI1-18	Channel No. 4 in (+)
5	EI1-19	Channel No. 4 in (-)
6	EO1-25	START from CIM (+)
7	EO1-24	START from CIM (-)
8	17-NOA	Signal No. 1 out (+)
9	17-NOB	Signal No. 1 out (-)
10	19-NOA	Signal No. 3 out (+)
11	19-NOB	Singal No. 3 out (-)
12	21-NOA	Signal No. 5 out (+)
13	21-NOB	Signal No. 5 out (-)
14	EI1-1	Channel No. 1 in (+)
15	EI1-2	Channel No. 1 in (-)
16	EI1-4	Channel No. 3 in (+)
17	EI1-5	Channel No. 3 in (-)
18	EI1-7	Channel No. 5 in (+)
19	EI1-8	Channel No. 5 in (-)
20	NC	
21	NC	
22	18-NOA	Signal No. 2 out (+)
23	18-NOB	Signal No. 2 out (-)
24	20-NOA	Signal No. 4 out (+)
25	20-NOB	Signal No. 4 out (-)

TABLE II. - AJ2 PIN CONNECTIONS -
PROTOBOARD SINGLE ENDED SIGNALS

Pin	To	Function
1	1-CA	NOISE No. 1
2	3-CA	NOISE No. 2
3	5-CA	NOISE No. 5
4	NC	
5	DJ1-16	BIAS No. 1
6	DJ1-12	BIAS No. 3
7	DJ1-1	Bias No. 5
8	MJ2-23	SCALE No. 2 to adder
9	MJ3-23	SCALE No. 4 to adder
10	NC	
11	MJ1-16	SCALE No. 1 from instr. amp
12	MJ2-16	SCALE No. 3 from instr. amp
13	MJ3-16	SCALE No. 5 from instr. amp
14	2-CA	NOISE No. 2
15	4-CA	NOISE No. 4
16	BNC (-)	NOISE GROUND
17	DJ1-14	BIAS No. 2
18	DJ1-2	BIAS No. 4
19	NC	
20	MJ1-17	SCALE No. 1 to adder
21	MJ2-17	SCALE No. 3 to adder
22	MJ3-17	SCALE No. 5 to adder
23	NC	
24	MJ2-22	SCALE No. 2 from instr. amp
25	MJ3-22	SCALE No. 4 from instr. amp

TABLE III. - MJ1, MJ2, and MJ3 PIN CONNECTIONS - MDAC (DAC-02) SCALE FACTOR SIGNALS

Connector MJ1		
Pin	To	Function
1-15	NC	
16	AJ2-11	SCALE No. 1 input to MDAC
17	AJ2-20	SCALE No. 1 output from MDAC
18-21	NC	
22	BNC (+)	NOISE Signal in
23	1-NOA, 2-NOA 3-NOA, 4-NOA, 5-NOA	NOISE Signal out
24-25	NC	

Connector MJ2		
Pin	To	Function
1-15	NC	
16	AJ2-12	SCALE No. 3 input to MDAC
17	AJ2-21	SCALE No. 3 output from MDAC
18-21	NC	
22	AJ2-24	SCALE No. 2 input to MDAC
23	AJ2-8	SCALE No. 2 output from MDAC
24-25	NC	

Connector MJ3		
Pin	To	Function
1-15	NC	
16	AJ2-13	SCALE No. 5 input to MDAC
17	AJ2-22	SCALE No. 5 output from MDAC
18-21	NC	
22	AJ2-25	SCALE No. 4 input to MDAC
23	AJ2-9	SCALE No. 4 output from MDAC
24-25	NC	

TABLE IV. - EI1 PIN CONNECTIONS -
ENGINE INPUT SIGNALS

Pin	To	Function
1	17-NCA, AJ1-14	Channel No. 1 in (+)
2	17-NCB, AJ1-15	Channel No. 1 in (-)
3	E01-3	Channel No. 2 shield
4	19-NCA, AJ1-16	Channel No. 3 in (+)
5	19-NCB, AJ1-17	Channel No. 3 in (-)
6	E01-6	Channel No. 4 shield
7	21-NCA, AJ1-18	Channel No. 5 in (+)
8	21-NCB, AJ1-19	Channel No. 5 in (-)
9	NC	
10	NC	
11	NC	
12	NC	
13	NC	
14	E01-14	Channel No. 1 shield
15	18-NCA, AJ1-2	Channel No. 2 in (+)
16	18-NCB, AJ1-3	Channel No. 2 in (-)
17	E01-17	Channel No. 3 shield
18	20-NCA, AJ1-4	Channel No. 4 in (+)
19	20-NCB, AJ1-5	Channel No. 4 in (-)
29	E01-20	Channel No. 5 shield
21	NC	
22	NC	
23	NC	
24	NC	
25	NC	

TABLE V. - DJ1 PIN CONNECTIONS -
DAC (DDA-06) OUTPUT SIGNALS

Pin	To	Function
1	AJ2-7	BIAS No. 5
2	AJ2-18	BIAS No. 4
3	E01-24	ACK to CIM
4	ERB-4	To ERB relay board
5	ERB-5	
6	ERB-6	
7	ERB-7	
8	ERB-8	
9	ERB-9	
10	ERB-10	
11	ERB-11	To ERB relay board
12	AJ2-6	Bias No. 3
13	NC	
14	AJ2-17	Bias No. 2
15	NC	
16	AJ2-5	Bias No. 1
17	NC	
18	NC	
19	NC	
20	NC	
21	NC	
22	AJ1-1	CIM Ready
23	ERB-23	To ERB relay board
24	ERB-24	
25	ERB-25	
26	ERB-26	
27	ERB-27	
28	ERB-28	
29	ERB-29	
30	ERB-30	
31	ERB-31	
32	ERB-32	
33	ERB-33	
34	ERB-34	
35	ERB-35	
36	ERB-36	
37	ERB-37	To ERB relay board

TABLE VI. - EO1 PIN COONECTIONS -
SIMULATOR OUTPUT SIGNALS

Pin	To	Function
1	17-CA	Channel No. 1 out (+)
2	17-CB	Channel No. 1 out (-)
3	EI1-3	Channel No. 2 shield
4	19-CA	Channel No. 3 out (+)
5	19-CB	Channel No. 3 out (-)
6	EI1-6	Channel No. 4 shield
7	21-CA	Channel No. 5 out (+)
8	21-CB	Channel No. 5 out (-)
9	NC	
10	NC	
11	NC	
12	NC	
13	NC	
14	EI1-14	Channel No. 1 shield
15	18-CA	Channel No. 2 out (+)
16	18-CB	Channel No. 2 out (-)
17	EI1-17	Cahnnel No. 3 shield
18	20-CA	Channel No. 4 out (+)
19	20-CB	Chaneel No. 4 out (-)
20	EI1-20	Channel No. 5 shield
21	NC	
22	NC	
23	NC	
24	AJ1-7	START from CIM (-)
25	AJ1-6	START from CIM (+)

BNC connector (noise input)

(+) - MJ1-22
(-) - 1-NCA, 2-NCA, 3-NCA
4-NCA, 5-NCA, AJ2-16
MJI-2

TABLE VII. - PROGRAM DESCRIPTIONS FOR THE SENSOR FAILURE SIMULATOR PROGRAM

Program name	DOS file name	Description
Cim	CIM.ASM	FORTRAN callable assembly routine used by SFS to begin failure simulation on cue from CIM unit.
Edit	EDIT.FOR	FORTRAN subroutine which controls flow of the failure scenario editor.
Edit Description	EDITDESC.FOR	FORTRAN subroutine which allows user to change description of current failure scenario.
Edit Failures	EDITFAIL.FOR	FORTRAN subroutine which allows user to define which channels of the current scenario will be failed.
Edit Gains	EDITGAIN.FOR	FORTRAN subroutine which allows user to define type of failure and associated constants for each failed channel.
Edit Save	EDITSAVE.FOR	FORTRAN subroutine which allows the user to save the current failure scenario in a DOS text file before existing failure scenario editor.
Edit Values	EDITVALU.FOR	FORTRAN subroutine which allows user to modify nominal and maximum channel values as well as failure delays for the current scenario.
End of Fil	ENDOFFIL.FOR	FORTRAN logical function which sets the pointer to the end of the currently open scenario file. True is returned if no errors, otherwise false is returned.
Erase Screen	ERASESCR.FOR	FORTRAN subroutine which writes the DOS control code for clearing the CRT text screen.
Get Clock	GETCLOCK.ASM	FORTRAN callable assembly routine which reads the real-time clock on the AST board. Returns as integers minutes, seconds, tenths, hundredths, and thousandths.
GETDAT		Non-standard FORTRAN callable subroutine whcih returns the date from the DOS clock.
GETTIM		Non-standard FORTRAN callable subroutine which returns the time from the DOS clock.
Init 8255	INIT82255.ASM	FORTRAN callable assembly routine which initializes the 8255 chip on the Metrabyte DDA-06 DAC/parallel output board. (Ports A and B initialized as outputs, port C initialized as input.).

TABLE VII. Continued

Program name	DOS file name	Description
Init HDWR	INITHDWR.FOR	FORTRAN subroutine which prepares failure hardware for transient simulation. Sets scale factor to 1.0 and bias to 0.0 then trips relays of failed channels.
Initialize	INITIAL.FOR	FORTRAN subroutine which initailizes values for all common blocks. Also calls subroutines which initialize the anlaog hardware (Init 8255 and Init HDWR).
Instructions	INSTRUCT.FOR	FORTRAN subroutine which calls READ INSTRUCTIONS to read DOS text file containing instructions (\SFS\INSTRUCT.TXT) then displays the file contents on the monitor in an organized fashion.
Menu 1	MENU1.FOR	FORTRAN subroutine which displays SFS main menu, prompts user and returns a correct response to the main program.
Menu 2	MENU2.FOR	FORTRAN subroutine which displays menu controlling manipulation of stored scenarios, prompts user and returns a correct response to STORE.
Noise CHK	NOISECHK.FOR	FORTRAN subroutine which checks failure scenario to ensure that noise is defined for one channel only. If not prompts user for correction.
NonBlank	NONBLANK.FOR	FORTRAN integer function whose value is the position of the last nonblank character in the character variable which is its parameter.
Open File	OPENFILE.FOR	FORTRAN subroutine used to open failure scenario files for reading and writing.
Read Instructions	READINST.FOR	FORTRAN subroutine which reads DOS text file containing insturctions for using the SFS.
Read Scenario	READSCE.FOR	FORTRAN subroutine which reads a scenario (specified by number) from the currently open scenario file.
Read Timer	ZEROTIME.FOR	FORTRAN subroutine which calls GET CLOCK to read the real-time clock on the AST board and returns as the value of its parameter the elaspsed time (in seconds) since the last call to ZERO TIMER. Resolution: 0.001 sec.
Run	RUN.FOR	FORTRAN subroutine which controls flow of program's real-time failure simulation.

TABLE VII. Concluded

Program name	DOS file name	Description
Run Reset	RUNRESET.FOR	FORTRAN subroutine which forces user to manually reset relays for all failed channels to an unfailed state at the end of failure simulation.
Run Setup	RUNSETUP.FOR	FORTRAN subroutine which prompts user to initialized failure hardware. If positive response, calls INIT HDWR.
Sensor Failure	SFS.FOR	FORTRAN MAIN PROGRAM. Controls overall program execution.
SFS Out	SFSOUT.ASM	FORTRAN callable assembly routine used for ouput to D/A converters, multiplying D/A converters, and relays located on the Metrabyte DDA-06, DAC-02, and ERB-24 boards respectively.
Store	STORE.FOR	FORTRAN subroutine which controls flow for portion of program which stores, retrieves, deletes, and replaces scenarios in DOS text files.
Store Delete	STOREDEL.FOR	FORTRAN subroutine which provides capability for user to delete a previously stored scenario from a file.
Store List	STORELIS.FOR	FORTRAN subroutine which reads a specified scenario file, presents user with a list of scenarios stored therein, prompts user for choise and returns number of the stored scenario to the calling subroutine.
Store Replace	STOREREP.FOR	FORTRAN subroutine which provides capability to replace any given scenario stored in a file with the current failure scenario.
Store Retrieve	STORERET.FOR	FORTRAN subroutine which provides capability for user to retrieve from a file a previously stored scenario.
Store Save	STORESAV.FOR	FORTRAN subroutine which provides a capability for the user to save the current failure scenario in a DOS text file.
Test	TEST.FOR	FORTRAN subroutine which allows user to manipulate the failure hardware directly from the PC's keyboard.
Wait	WAIT.FOR	FORTRAN subroutine which uses GETTIM to suspend program execution for a specified number of seconds.
Write Scenario	WRITESCE.FOR	FORTRAN subroutine which writes the current failure scenario to the currently open scenario file.
Zero Timer	ZEROTIME.FOR	FORTRAN subroutine which calls GET CLOCK to read the real-time clock on AST board then stores the time of day returned by GET CLOCK as the start time of the transient.

TABLE VIII. - COMMON BLOCK DESCRIPTIONS FOR THE SENSOR FAILURE SIMULATOR PROGRAM

Name of common	DOS file name	Type	Description
(Blank)	BLANK.CMN	character	Contains DOS control codes and special ASCII graphics sequences.
SFS 00	SFS00.CMN	character	Contains name of default scenario file, file header, current scenario description and units for each channel.
SFS 01	SFS01.CMN	integer	Contains constants for number of channels, number of failures per channel and maximum number of scenarios per file.
SFS 02	SFS02.CMN	mixed	Contains logical and numerical data for the current failure scenario.
Menu 00	MENU00.CMN	character	Contains title and options used by subroutine menu 1.
Menu 11	MENU11.CMN	character	Contains description of channel signals for editor.
Menu 12	MENU12.CMN	character	Contains description of failure modes for editor.
Menu 21	MENU21.CMN	character	Contains description of options for the stored scenario menu.
Hardware	HARDWARE.CMN	integer	Contains device designations for the various pieces of D/A hardware.

TABLE IX. - HIERARCHICAL STRUCTURE OF THE SENSOR FAILURE SIMULATOR PROGRAN

Main Program	Level 1	Level 2	Level 3	Level 4	Level 5
Sensor Failure Simulator	Initialize	Init 8255			
		Init HDWR	SFS Out		
	Erase Screen				
	Menu 1	Erase Screen			
		NonBlank			
	Instructions	Read Instructions			
		Erase Screen			
	Store	Erase Screen			
		Menu 2			
		Store Retrieve	Open File	NonBlank	
			Store List		
			Read Scenario	NonBlank	
				Wait	GETTIM
		Store delete	Open File	NonBlank	
			Store List		
			NonBlank		
		Store Replace	Open File	NonBlank	
			Store List		
			Write Scenario	GETDAT	
				GETTIM	
				NonBlank	
			NonBlank		
		Store Save	Open File	NonBlank	
			End of File		
			Write Scenario	GETDAT	
				GETTIM	
				NonBlank	
			NonBlank		
	Edit	Erase Screen			
		Edit Description	NonBlank		
		Edit Failure			

TABLE IX. - HIERARCHICAL STRUCTURE OF THE SENSOR FAILURE SIMULATOR PROGRAN
(concluded)

Main Program	Level 1	Level 2	Level 3	Level 4	Level 5
		Edit Values			
		Edit Gains			
		Noise Check			
		Edit Save	Open File	NonBlank	
			End of File		
			Write Scenario	GETDAT	
				GETTIM	
				NonBlank	
			NonBlank		
			Wait	GETTIM	
	Run	Erase Screen			
		Run Setup	Init HDWR	SFS Out	
		CIM			
		Zero Timer	Get Clock		
		Read Timer	Get Clock		
		(Zero Timer)			
		SFS Out			
		Run Reset	SFS Out		
	Test	Erase Screen			
		SFS Out			

TABLE X. - EDITOR FUNCTION KEYS AND CODES

Key	Code	Function
F1	m1<CR>[a]	display scenario description menu
F2	m6<CR>	display failed channels menu
F3	m2<CR>	display nominal values menu
F4	m7<CR>	display maximum values menu
F5	m3<CR>	display failure delay menu
F6	m8<CR>	display channel 1 failure modes
F7	m4<CR>	display channel 2 failure modes
F8	m9<CR>	display channel 3 failure modes
F9	m5<CR>	display channel 4 failure modes
F10	m0<CR>	display channel 5 failure modes
Home	m0<CR>	display scenario description menu
PgUp	b<CR>	display previous menu
PgDn	f<CR>	display next menu
End	mx<CR>	save scenario and exit editor
↑	u<CR>	move up one menu item
↓	d<CR>	move down one menu item

[a]<CR> denotes a carriage return (ASCII decimal code 13)

TABLE XI. - DOS BATCH FILE SFS BAT

Line no.	DOS Command line
1	echo off
2	break off
3	path D:\sfs;D:\DOS;D:\utility
4	cd\scenario
5	if '%1'= ='/d'; dir *.sce > LPT1
6	if '%1'= ='/D'; dir *.sce > LPT1
7	SFS
8	cd\
9	break on

TABLE XII. - SPECIAL CODES FOR INSTRUCTION TEXT FILE

Code	Translation
$$	dollar sign ($)
$e	escape character, ASCII decimal code 27
$f	control sequence which causes subsequent test to flash (blink)
$b	control sequence which causes subsequent text to appear bold (high intensity)
$r	control sequence which causes subsequent text to appear in reverse video (dark on light)
$n	control sequence which causes subsequent text to appear in normal video (non-flashing, normal intensity, light on dark)
$c###	ASCII character specified by ###, where is a three digit integer between 000 and 255

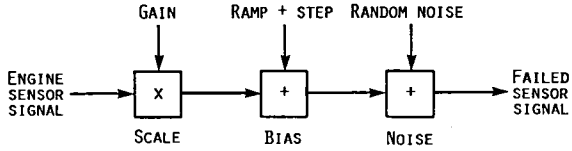

FIGURE 1. - METHOD OF FAILURE MODELING.

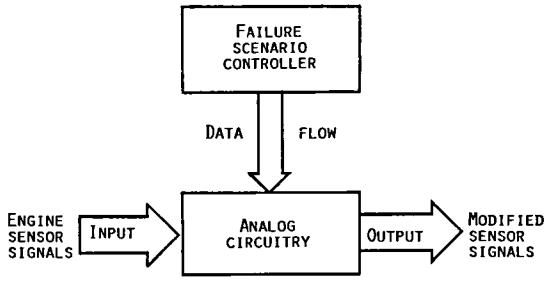

FIGURE 2. - GENERAL CONCEPTUAL DESIGN.

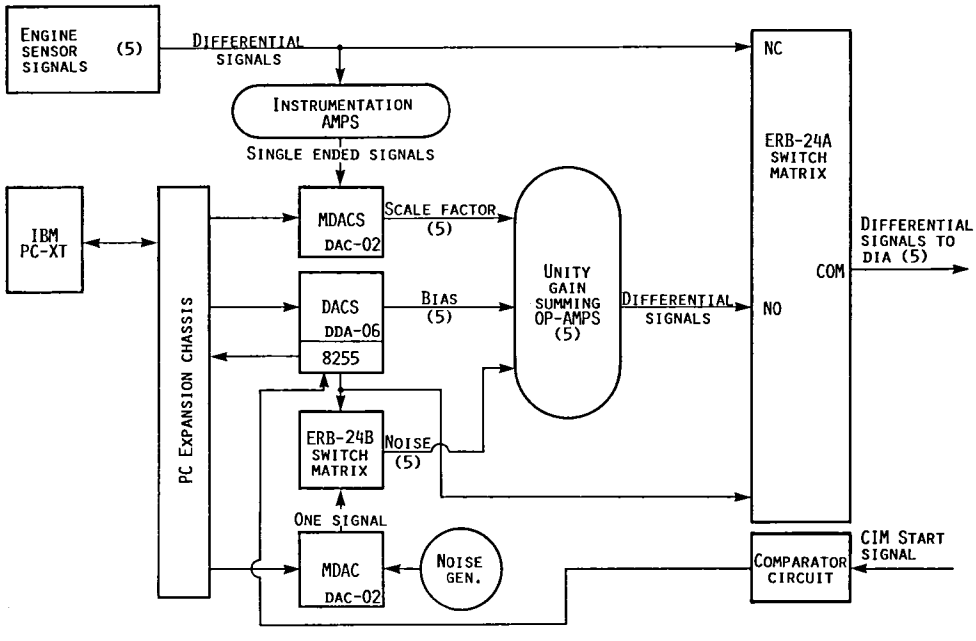

FIGURE 3. - BLOCK DIAGRAM OF SFS HARDWARE DESIGN.

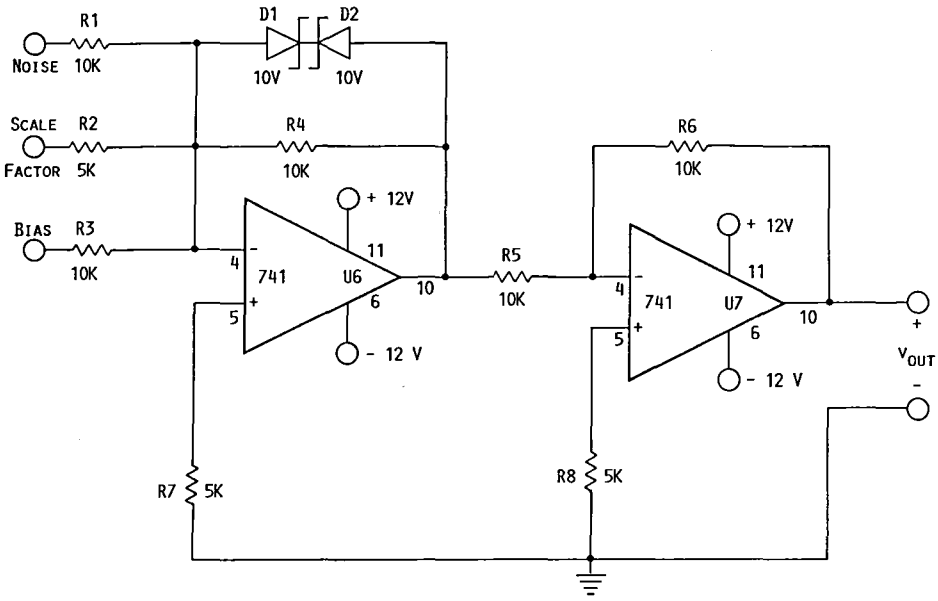

FIGURE 4. - TYPICAL (CHANNEL 1) CLIPPING-SUMMING AMPLIFIER.

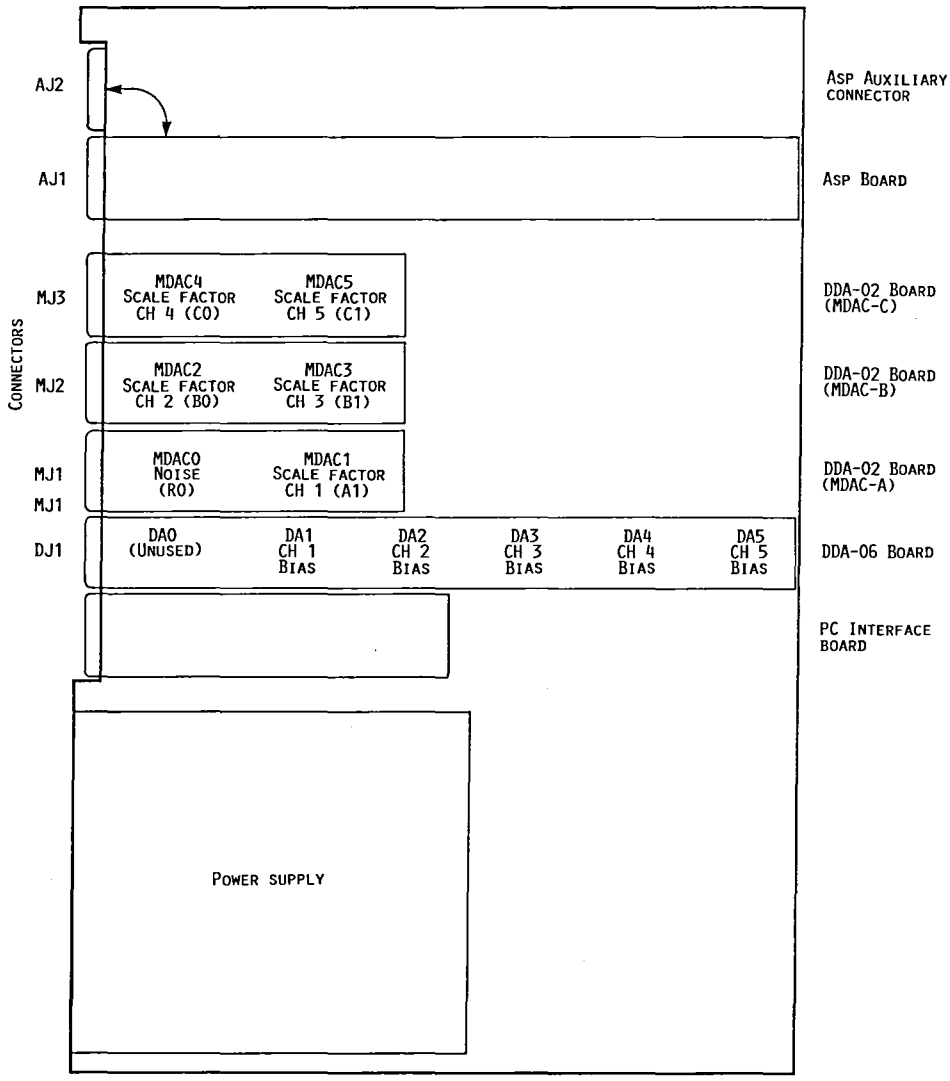

FIGURE 5. - PC EXPANSION CHASSIS LAYOUT.

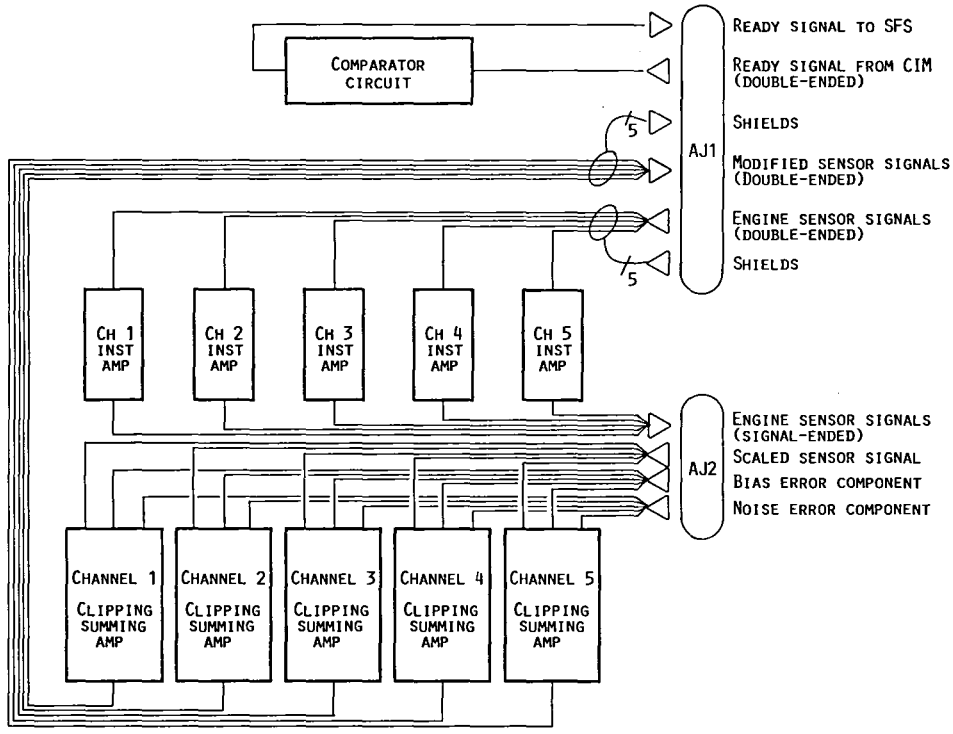

FIGURE 6. - BLOCK DIAGRAM OF ASP BOARD.

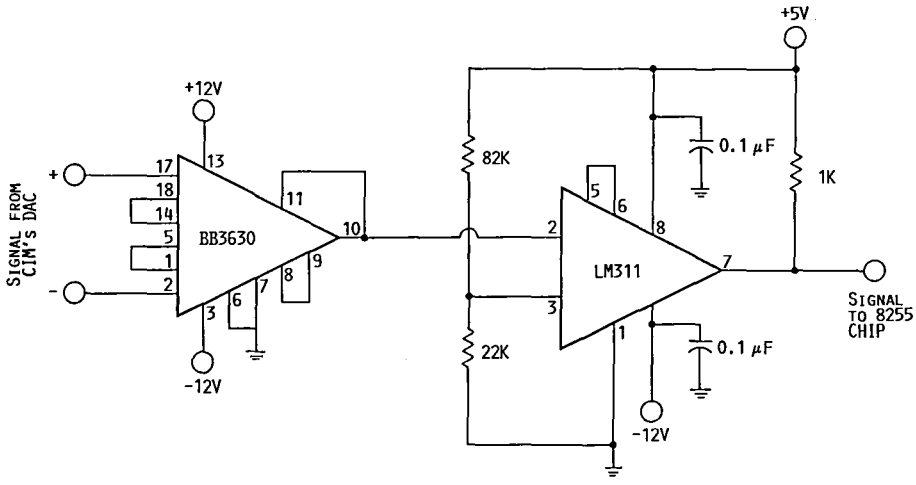

FIGURE 7. - COMPARATOR CIRCUIT FOR CIM/PROTOBOARD COMMUNICATIONS.

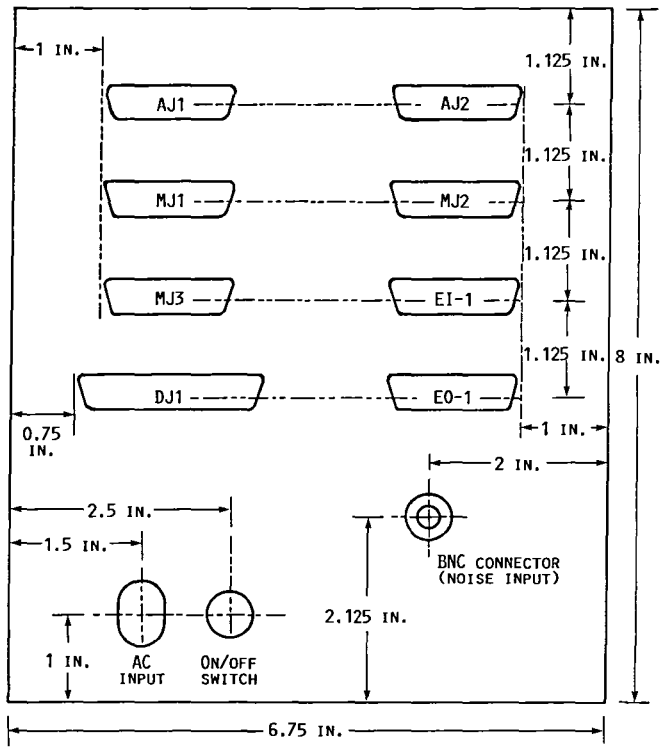

FIGURE 8. - ERB CHASSIS END PANEL.

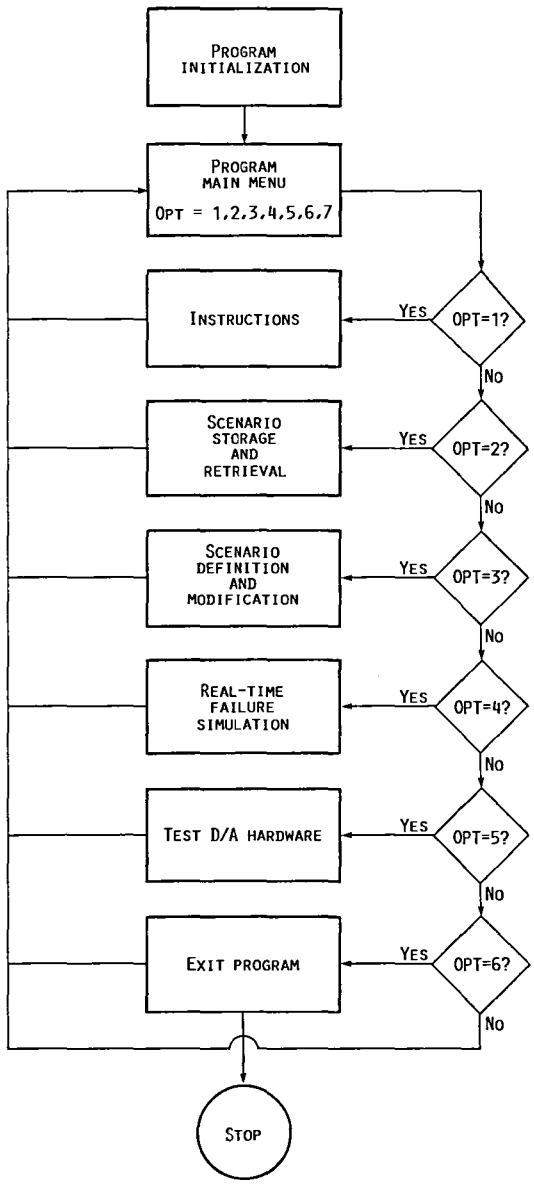

FIGURE 9. - GENERAL CONCEPTUAL DESIGN OF SFS SOFTWARE.

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                       *
*                  Sensor Failure Simulator             *
*                                                       *
*                        MAIN MENU                      *
*                                                       *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * *

      1)   INSTRUCTIONS

      2)   RETRIEVE Stored  Failure Scenario
      3)   EDIT ... Current Failure Scenario
      4)   Run .... Current Failure Scenario

      5)   TEST ... SFS Hardware/Software

      6)   QUIT

YOUR CHOICE?
```

FIGURE 10. - SENSOR FAILURE SIMULATOR MAIN MENU.

```
                HOW TO USE THESE INSTRUCTIONS


     Welcome to the Sensor Failure Simulator!  You have just accessed
the on-line instruction file.  This file describes the operation of the
Sensor Failure Simulator (refered to as the SFS in the remaining pages
of this documentation).

     These instructions consist of a number of pages of text.  You may
page up and down through the text by pressing the PgUp and PgDn keys.
To return to this menu, press the Home key.  To view the last page of
this text press the End key.  You may exit the on-line instruction mode
at any time by pressing the carriage return/enter key.  If later you
decide to return for more instruction, the program will automatically
begin with the page at which instruction was previouly terminated.
Enter a PgDn to proceed to the introduction.
```

FIGURE 11. - FIRST PAGE OF ON-LINE INSTRUCTIONS.

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                               *
*                    Sensor Failure Simulator                   *
*                                                               *
*                      STORED SCENARIO MENU                     *
*                                                               *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

* *      Current Scenario Description Shown Here          * *


      1)   RETRIEVE Stored  Failure Scenario
      2)   DELETE   Stored  Failure Scenario
      3)   REPLACE  Stored  Failure Scenario
      4)   STORE    Current Failure Scenario

      5)   Return to Main Menu

  YOUR CHOICE?
```

FIGURE 12. - SFS STORED SCENARIO MENU.

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                               *
*                    Sensor Failure Simulator                   *
*                                                               *
*                      STORED SCENARIO MENU                     *
*                                                               *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

* *      Current Scenario Description Shown Here          * *

   0) N1 Step Failure:  1000 rpm @ 0.5 sec          22NOV85
   1) N1 Ramp Failure:   200 rpm/sec @ 1.0 sec      22NOV85
   2) N2 Step Failure:  1500 rpm @ 1.0 sec          25NOV85
   3) N2 Ramp Failure:   250 rpm/sec @ 1.5 sec      25NOV85
   4) N3 Step Failure:  1000 rpm @ 0.5 sec          29NOV85
   5) N3 Ramp Failure:   200 rpm/sec @ 1.0 sec      29NOV85
   6) N4 Step Failure:  1500 rpm @ 1.0 sec          05DEC85
   7) N4 Ramp Failure:   250 rpm/sec @ 1.5 sec      05DEC85
   8) N5 Step Failure:  1500 rpm @ 0.5 sec          10DEC85
   9) N5 Ramp Failure:   250 rpm/sec @ 1.0 sec      10DEC85

Enter:  Value  or  ◄┘:
```

FIGURE 13. - TYPICAL MENU OF STORED SCENARIO DESCRIPTIONS.

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                               *
*                    Sensor Failure Simulator                   *
*                                                               *
*                    FAILURE SCENARIO EDITOR                    *
*                                                               *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

    Enter A 45 Character Description Of The Scenario:

    0   5   10   15   20   25   30   35   40   45
     Sensor Failure Simulator Default Scenario
```

FIGURE 14. - ENTERING THE FAILURE SCENARIO EDITOR.

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                               *
*                    Sensor Failure Simulator                   *
*                                                               *
*                    FAILURE SCENARIO EDITOR                    *
*                                                               *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

* *      Sensor Failure Simulator Default Scenario        * *

  Channels To Be Failed Are Displayed IN Bold Type:

    Channel 1)  Low  Spool  Shaft Speed
    Channel 2)  High Spool  Shaft Speed
    Channel 3)  Combustor   Exit  Pressure
    Channel 4)  Low turbine Exit  Pressure
    Channel 5)  Low Turbine Inlet Temperature

  Toggle On/Off By Channel Number or ◄┘:
```

FIGURE 15. - FAILED CHANNELS MENU.

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                               *
*                    Sensor Failure Simulator                   *
*                                                               *
*                    FAILURE SCENARIO EDITOR                    *
*                                                               *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

* *      Sensor Failure Simulator Default Scenario        * *

Define The Nominal Value For Each Failed Channel:

  Channel 1: Low  Spool  Shaft Speed       =  10000.00 RPM*
  Channel 2: High Spool  Shaft Speed       =  13000.00 RPM
  Channel 3: Combustor   Exit  Pressure    =  400.0000 PSI
  Channel 4: Low Turbine Exit  Pressure    =  50.00000 PSI
  Channel 5: Low Turbine Inlet Temperature =  1700.000  °F

Enter:  Value  or  ◄┘:
```

FIGURE 16. - MENU FOR NOMINAL CHANNEL VALUES.
*VALUES SHOWN ARE FOR ILLUSTRATION ONLY.

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                           *
*                  Sensor Failure Simulator                 *
*                                                           *
*                  FAILURE SCENARIO EDITOR                  *
*                                                           *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

* *      Sensor Failure Simulator Default Scenario     * *

Define The Maximum Value For Each Failed Channel:

  Channel 1: Low  Spool  Shaft Speed       =  15000.00 RPM*
  Channel 2: High Spool  Shaft Speed       =  15000.00 RPM
  Channel 3: Combustor   Exit  Pressure    =  600.0000 PSI
  Channel 4: Low Turbine Exit  Pressure    =  100.0000 PSI
  Channel 5: Low Turbine Inlet Temperature =  2500.000  °F

Enter:  Value  or  ◄┘:
```

FIGURE 17. - MENU FOR MAXIMUM CHANNEL VALUES.
*VALUES SHOWN ARE FOR ILLUSTRATION ONLY.

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                           *
*                  Sensor Failure Simulator                 *
*                                                           *
*                  FAILURE SCENARIO EDITOR                  *
*                                                           *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

* *      Sensor Failure Simulator Default Scenario     * *

Define The Failure Delay For Each Failed Channel:

  Channel 1: Low  Spool  Shaft Speed       =  0.00000E+00 SEC
  Channel 2: High Spool  Shaft Speed       =  0.00000E+00 SEC
  Channel 3: Combustor   Exit  Pressure    =  0.00000E+00 SEC
  Channel 4: Low Turbine Exit  Pressure    =  0.00000E+00 SEC
  Channel 5: Low Turbine Inlet Temperature =  0.00000E+00 SEC

Enter:  Value  or  ◄┘:
```

FIGURE 18. - FAILURE DELAY MENU.

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                           *
*                  Sensor Failure Simulator                 *
*                                                           *
*                  FAILURE SCENARIO EDITOR                  *
*                                                           *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

* *      Sensor Failure Simulator Default Scenario     * *

     Channel 1 Active Failures Are In Bold Type:

        1)  Scale Factor = [    1.000000      ]
        2)  Bias         = [  0.0000000E+00 RPM ]
        3)  Ramp         = [  0.0000000E+00 RPM/sec ]
        4)  Noise S.F.   = [  0.0000000E+00 ]

     Toggle On/Off By Number [,Value] or  ◄┘:
```

FIGURE 19. - TYPICAL MENU OF FAILURE MODES AND CONSTANTS.

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                           *
*                  Sensor Failure Simulator                 *
*                                                           *
*                  FAILURE SCENARIO EDITOR                  *
*                                                           *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

* *      Sensor Failure Simulator Default Scenario     * *

          _____ Based On Failure Gains _____

     Channel Value Will Exceed User Defined Limits:

        A) User Defined Maximum =     15000.00
        B) Calculated   Maximum =     15501.00

     Change "A" to "B" ?  Y
```

FIGURE 20. - MAXIMUM CHANNEL VALUE EXCEEDED.

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                           *
*                  Sensor Failure Simulator                 *
*                                                           *
*                  FAILURE SCENARIO EDITOR                  *
*                                                           *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

* *      Sensor Failure Simulator Default Scenario     * *

          ____ Ramp Failure Will Peak At Approx. ____

                        0 minutes
                        8 seconds

                  Is This Acceptable?  Y
```

FIGURE 21. - TYPICAL DISPLAY FOR ACTIVE RAMP FAILURE.

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                           *
*                  Sensor Failure Simulator                 *
*                                                           *
*                  FAILURE SCENARIO EDITOR                  *
*                                                           *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

* *      Sensor Failure Simulator Default Scenario     * *

          Noise Is Allowed On One Channel Only.
          It Has Been Defined On 3 Channels:

                        Channel 1
                        Channel 2
                        Channel 4

          Which One Channel Should Be Failed?
```

FIGURE 22. - TYPICAL DISPLAY FOR NOISE FAILURE ON MULTIPLE CHANNELS.

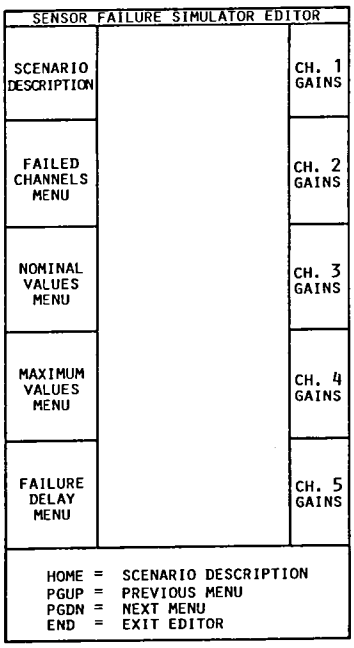

FIGURE 23. - SFS FUNCTION KEY TEMPLATE.

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                               *
*                  Sensor Failure Simulator                     *
*                                                               *
*                  RUNNING FAILURE SCENARIO                     *
*                                                               *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

* *      Sensor Failure Simulator Default Scenario      * *


    Initialize Sensor Failure Hardware?  Y
```

FIGURE 24. - QUERY TO INITIALIZE FAILURE HARDWARE.

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                               *
*                  Sensor Failure Simulator                     *
*                                                               *
*                  RUNNING FAILURE SCENARIO                     *
*                                                               *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

* *      Sensor Failure Simulator Default Scenario      * *


  How Many Seconds Should The Scenario Run?
              <default=20.00>
```

FIGURE 25. - QUERY FOR LENGTH OF SIMULATION.

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                               *
*                  Sensor Failure Simulator                     *
*                                                               *
*                  RUNNING FAILURE SCENARIO                     *
*                                                               *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

* *      Sensor Failure Simulator Default Scenario      * *


      Ready To Begin Failure Scenario:

         1) Begin Scenario
         2) Begin Scenario on signal from CIM

         3) Return To Main Menu

      Enter Run Option:
```

FIGURE 26. - QUERY FOR SIMULATION START SIGNAL.

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                               *
*                  Sensor Failure Simulator                     *
*                                                               *
*                  RUNNING FAILURE SCENARIO                     *
*                                                               *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

* *      Sensor Failure Simulator Default Scenario      * *


                    ** RUNNING **
```

FIGURE 27. - SIGNAL FOR EXECUTION OF USER INITIATED REAL-TIME FAILURE SIMULATION.

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                               *
*                  Sensor Failure Simulator                     *
*                                                               *
*                  RUNNING FAILURE SCENARIO                     *
*                                                               *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

* *      Sensor Failure Simulator Default Scenario      * *


          _____ Waiting For Signal From CIM ____

          Star "*" Appears When Transient Begins

                              *
```

FIGURE 28. - SIGNAL FOR CIM INITIATED REAL-TIME FAILURE SIMULATION.

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                           *
*                  Sensor Failure Simulator                 *
*                                                           *
*                  RUNNING FAILURE SCENARIO                 *
*                                                           *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

* *      Sensor Failure Simulator Default Scenario     * *


      _____ Finished Running at 20.000 seconds _____


         Maximum Delta T is    0.002 seconds.
              Maximum Delta T at  19.997 seconds.
         Average Delta T is    0.001 seconds.


            *** HIT ◄┘ TO CONTINUE ***
```

FIGURE 29. - DISPLAY OF RUN TIME STATISTICS.

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                           *
*                  Sensor Failure Simulator                 *
*                                                           *
*                  RUNNING FAILURE SCENARIO                 *
*                                                           *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

* *      Sensor Failure Simulator Default Scenario     * *

  Manual Reset of Failed Channels ( Bold Typed ):

    Channel 1)  Low  Spool  Shaft Speed
    Channel 2)  High Spool  Shaft Speed
    Channel 3)  Combustor   Exit  Pressure
    Channel 4)  Low turbine Exit  Pressure
    Channel 5)  Low Turbine Inlet  Temperature

  Toggle Off By Channel Number:
```

FIGURE 30. - TYPICAL MENU FOR MANUAL RESET OF FAILED CHANNELS.

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                       *
*                Sensor Failure Simulator               *
*                                                       *
*                      TEST MENU                        *
*                                                       *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * *

    0) Noise DAC-02/MDAC0
    1) Scale DAC-02/MDAC1     11) Noise   Relay 1/PA1
    2) Scale DAC-02/MDAC2     12) Noise   Relay 2/PA2
    3) Scale DAC-02/MDAC3     13) Noise   Relay 3/PA3
    4) Scale DAC-02/MDAC4     14) Noise   Relay 4/PA4
    5) Scale DAC-02/MDAC5     15) Noise   Relay 5/PB5
    6) Bias  DDA-06/DA1       16) Failure Relay 1/PB1
    7) Bias  DDA-06/DA2       17) Failure Relay 2/PB2
    8) Bias  DDA-06/DA3       18) Failure Relay 3/PB3
    9) Bias  DDA-06/DA4       19) Failure Relay 4/PB4
   10) Bias  DDA-06/DA5       20) Failure Relay 5/PB5

  Enter Number To Toggle (exit=99) :
```

FIGURE 31. - TYPICAL MENU FOR MANUALLY CONTROLLING HARDWARE.

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA TM-87271		

4. Title and Subtitle	5. Report Date
A Sensor Failure Simulator For Control System Reliability Studies	July 1986
	6. Performing Organization Code 505-62-01

7. Author(s)	8. Performing Organization Report No.
Kevin J. Melcher, John C. Delaat, Walter C. Merrill Lawrence G. Oberle, and Gerald G. Sadler and Joseph H. Schaefer	E-3137
	10. Work Unit No.

9. Performing Organization Name and Address	11. Contract or Grant No.
National Aeronautics and Space Administration Lewis Research Center Cleveland, Ohio 44135	
	13. Type of Report and Period Covered Technical Memorandum

12. Sponsoring Agency Name and Address	14. Sponsoring Agency Code
National Aeronautics and Space Administration Washington, D.C. 20546	

15. Supplementary Notes

Kevin J. Melcher, John C. Delaat, Walter C. Merrill, Lawrence G. Oberle, and Gerald G. Sadler, NASA Lewis Research Center, Joseph H. Schaefer, United States Corps of Cadets, West Point, New York.

16. Abstract

A real-time Sensor Failure Simulator (SFS) was designed and assembled for the Advanced Detection, Isolation, and Accommodation (ADIA) program. Various designs were considered. The design chosen features an IBM-PC/XT. The PC is used to drive analog circuitry for simulating sensor failures in real-time. A user defined scenario describes the failure simulation for each of the five incoming sensor signals. Capabilities exist for editing, saving, and retrieving the failure scenarios. The (SFS) has been tested closed-loop with the Controls Interface and Monitoring (CIM) unit, the ADIA control, and a real-time F100 hybrid simulation. From a productivity viewpoint, the menu driven user interface has proven to be efficient and easy to use. From a real-time viewpoint, the software controlling the simulation loop executes at greater than 100 cycles/sec.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
Personal computer Analog electronics Fortran Failure simulation	Unclassified - unlimited STAR Category 33

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of pages	22. Price*
Unclassified	Unclassified		

*For sale by the National Technical Information Service, Springfield, Virginia 22161

End of Document